ADAPTIVESCREEN: AN ADAPTIVE NEWS BROWSER FOR LAPTOPS AND MOBILE DEVICES

BY

BOFAN SUN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

Adviser:

Professor Chengxiang Zhai

ABSTRACT

Optimizing interactive information retrieval interfaces is a new trend in IR which focuses more on the interface design and formalizes interactive IR. However, none of existing work has proposed a system to support both adaptive web design and navigational interface, a primary goal of this thesis. We propose the very first browsing system that can adapt to screen size and inferred user need potentially during the process of information retrieval at every interaction. AdaptiveScreen not only presents a user-friendly interface with adaptive web design but also connects with a novel interface card model which formally models the interactive retrieval task. We show that AdaptiveScreen improves upon the prototype system from the perspective of system architecture and system implementations. AdaptiveScreen is redesigned in the manner of classic software architectural pattern Model-View-Controller. By comparing the screen shots and performance between AdaptiveScreen and the prototype system on both laptops and movable devices, we can conclude that AdaptiveScreen successfully leverages the effectiveness of Interface Card Model (ICM) proposed before and overcomes the weakness of prototype system. In the end, we hope the new system can demonstrate the potential applications of algorithms based on ICM and stimulate other researchers in the field of interactive IR.

To my parents, for their love and support.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Most of Information retrieval (IR) researches aim to develop formal models. Among these studies, the Probability Ranking Principle (PRP) is one of the fundamental rule. It transforms the retrieval task to a ranking problem so that lots of state-of-the-art retrieval functions are developed based on this rule. However, some researchers [1, 2, 3] question that some assumptions hold by PRP, i.e., sequential browsing and independent relevance of documents, are not necessarily true in the real scenario. Among these researches, the authors of the PRP for interactive IR (IIR-PRP) are pioneers on addressing the independence assumption. However, they fail to undertake the sequential browsing assumption.

With the inspiration of the dynamic and interactive nature of information seeking process studied in [4, 5], the authors in the paper [1] attempt to build an intelligent IR system with an adaptive interface for navigation. As a result, they proposed a novel general formal model for optimizing interactive information retrieval interfaces called Interactive Card Model (ICM) which effectively covers the sequential browsing assumption. Furthermore, they designed a prototype interface to present their Navigation Card Model. Even though the prototype can effectively reflect the purpose of proposed models, after playing with the system, we still believe that it can be further improved from the perspective of adaptive web design. Therefore, we propose an adaptive news browser for mobile phones, AdaptiveScreen, which leverages the system architecture of previous prototype interface. AdaptiveScreen fully connects to the code of previous interface system and becomes the very first browsing system that can adapt to screen size and inferred user need potentially at every interaction. In addition, the new system can help demonstrate the potential applications of ICM.

This thesis is organized in the following way. Chapter 2 presents the related works which bring us the idea of designing an adaptive user interface for ICM.

Chapter 3 mainly introduces the system architecture in the manner of MVC and discusses the strength and weakness of the prototype system. Chapter 4 presents all major functions we developed for AdaptiveScreen and how we address challenges during the implementations. Chapter 5 compares the performance between the prototype and our new interface. In particular, we identify successful cases where the new system is better and unsuccessful cases where it might be worse. In Chapter 6, we summarize the functionality of AdaptiveScreen and suggest interesting directions for future research in the end.

CHAPTER 2

RELATED WORK

There are plenty of traditional retrieval models, such as vector space models [6], classic probabilistic models [7], and language models [8, 9, 10]. Most of these works are based on the Probability Ranking Principle (PRP) [11]. However, these works do not consider user interactions. With the development of information retrieval methods, some researchers show that user interface design can be involved into the retrieval process.

The authors of the PRP for interactive IR (IIR-PRP) [3] believes that interactive information retrieval (IIR) needs a broader view and should release some assumptions which the PRP often do not hold. They design a new theoretical framework for interactive retrieval which users can make decisions in each situations. Usually, the system will offer users a list of choice associated with a number of cost and probability parameters so that an optimum ordering of the choices can be derived. Our AdaptiveScreen system is based on Interface Card Model (ICM) [1], which shares a similar high-level goal in that both attempt to establish a formal model for interactive retrieval.

In ICM, the authors proposed a novel formal model for optimizing interactive retrieval interface to formally study the problem of automatic interface optimization. Besides the idea of considering users choice and cost, ICM designs the interactive retrieval process as a process of playing cooperative card game. In each interaction lap, the system aims to maximize the expected gain of relevant information for the users and in the meantime, minimize the effort of the users. Our new interface system, AdaptiveScreen, highly preserves the mechanism of ICM.

CHAPTER 3

SYSTEM ARCHITECTURE

AdaptiveScreen strictly follows the design of Navigational Card Model (NCM) which is an instantiation example of the ICM [1]. As an advanced version of previous interface systems prototype based on NCM, AdaptiveScreen not only leverages all existing features and functionality, but also supports adaptive web design which no existing method could achieve in a principled way. By completely analyzing the system architecture of previous prototype and developing new functions for the new interface, we successfully achieve the goal of developing the very first browsing system that can adapt to screen size and inferred user need potentially at every interaction. In this chapter, we introduce the details of system architecture in a manner of model, view and controller. Furthermore, all strength and weakness of the previous prototype interface are discussed at the end of each subsection.

3.1 Navigation Card Model (NCM)

NCM is an instantiation example of the ICM, which treats any user-interface interaction as a card game. Within the ICM, the interactive retrieval task is completed by the system and the user together. In each lap, player determines the optimal card to play and the system maximizes players benefits. By considering users behaviors, history of interaction, the reward and cost of users' next movements and limitations associated with the cards, the ICM will navigate users to their desired information. In summary, the ICM keeps the generality and releases the *sequential browsing* assumption which is lying beneath all other existing theoretical IR models [6, 7, 8, 9, 10, 12, 13]. In addition, the *sequential interaction* between the user and the interface system in each laps fits much better in the real world scenario. Since time only passes in the manner of uni-directionality, the temporal nature of *sequen-*

tial interaction can always be true and derived by IIR-PRP model proposed in [3].

In the Navigation Card Model, we offer both navigational elements and items themselves together on the interface. These navigational tags give user the ability to quickly find desired items by simply selecting these tags. However, users may be confused about the interface with both tags and items appeared, and therefore, lots of classic user interface design questions raise during our research. In the previous prototype, the system made default settings to determine how many tags and items showed on the interface, how to switch between tag panel and item panel, and how large the card would be in the screen. With the improvements on adaptive web design, AdaptiveScreen presents more user-friendly interfaces and can help to demonstrate the potential applications of NCM on the mobile platform.

In summary, Navigation Card Model is the central component of the pattern and directly manages the data, logic and rules of the application. The model is already well-defined and proved to be effective in the paper [1]. Therefore, AdaptiveScreen leverages the design of NCM without any modifications. Instead, the new system focuses on building a more user-friendly automatic layout adjustment interface.

3.2 Introduction of interface mechanism

The previous prototype interfaces were built on top to the set of most popular news articles and their associated keywords returned from the New York Times Most Popular API[1], in which the articles and the keywords respectively correspond to the items and the tags in the model. In specific, there are only two types of screen sizes supported, medium sized and very small sized screen, which are presented in Figure 3.1(a) and Figure 3.1(b). At the very first place, the algorithm wisely chooses to present only tags in the very small sized screen, but includes both items (articles showed on the left) and tags in the medium sized screen. The partitioning strategy is very reasonable because the system cannot anticipate which news articles the user is looking for. Hence, the action of showing plenty of tags instead of a single article can more likely satisfy the demand of user. In addition, in the very small sized

[1]http://developer.nytimes.com/

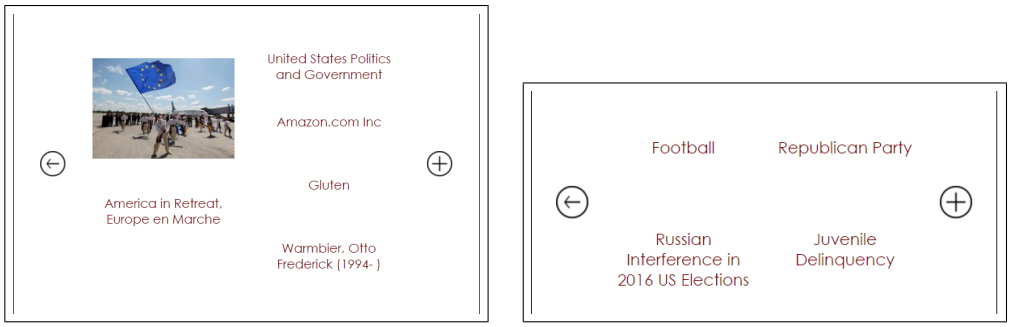

Figure 3.1: Prototype interface in (a) medium screen (b) small screen.

screen, whenever user selects *Football*, the interface will automatically adjust the layout and offer a specific news article related to *Football*. The mechanism behind the screen considers that only a few articles in the database are relevant to *Football*. Therefore, displaying an item directly is more beneficial to user in this scenario. Even the chosen article may not be interested by users, they still have the opportunity to select the plus button positioned on the right edge of the block which can lead them to next relevant news article. At the end, if users make any misoperations, we offer an arrow button placed on the left edge of the block which gives users the chance to go back to previous stage.

In conclusion, the overall performance of the previous prototype is highlighted by the effectiveness and generality over the pre-designed static interfaces in user studies. However, the interface system is designed in a non-responsive technique with only two types of sized screen exhibited. In addition, the design of this view leaves plenty of white space on the web page, which is a waste as an information retrieval system. Last but not least, users may expect to access this news browser through a mobile phone. In most cases, a mobile phone support a rotation features. The current interface has no reflection on that either.

3.3 Interaction between NCM and front-end mechanism

With the algorithm defined by NCM and the news articles crawled from the New York Times Most Popular API, we have the full management on the logic flow and data source. With the design of items block, tags block,

next button and back button, we provide users a decent and straight-forward front-end design. However, to achieve the goal of building a functional web application, we still miss the last piece of puzzle which is a controller sending commands to the model to update models state and reflecting corresponding changes into front-end interface.

At the very first place, the controller in previous prototype decrypts the website URL into 3 parts, and sends those information as three parameters to the server, including *size*, *app*, and *algo* respectively. In specific, *size* controls how big the interface screen will be displayed, which has only two options medium and very small. In addition, *app* refers to the source of data, with value such as Walmart, Wikipedia and New York Times. In the end, *algo* denotes which interface card model the server is playing. As claimed in paper [1], experimental results show that Navigational Card Model beats other baseline static interfaces. Therefore, AdaptiveScreen will always apply NCM as the *algo* parameter. In conclusion, these three parameters are the fundamentals for loading the initial screen of website.

After entering into the initial lap, users now face to the interface showed in Figure 3.1(a) or Figure 3.1(b). By using the very small sized screen in Figure 3.1(b) as an example, there are four types of clickable buttons. At the initial screen, the most essential buttons are four tag blocks which represent subsets of news articles. Behind the screen, they are labeled by the controller from 0 to 3 in the order of top left, top right, bottom left and bottom right respectively. Whenever users select any one of them, they can find three more parameters, *session*, *lap* and *selected* following predefined ids showed up in the URL. In details, *session* id will never change among the whole interactions unless users go back to the first lap. The number represents a set of data files crawling from the internet. In addition, the meanings of *lap* id and *selected* id are straight-forward. Any actions offering more information will be considered as next lap. For example, clicking tags or next button will make the *session* id increased by 1. In contrast, it will decrease by 1 until reaching the minimum value 0 if user selects back button. Please note that the initial screen does not contain any of these three parameters since the system decides to offer user random information in each browsing experience. For *selected* id, it denotes the positions of four tags so that the controller realizes which subsets of items to retrieve through server. Last but not least, with the help of Navigation Card Model, most of users will reach

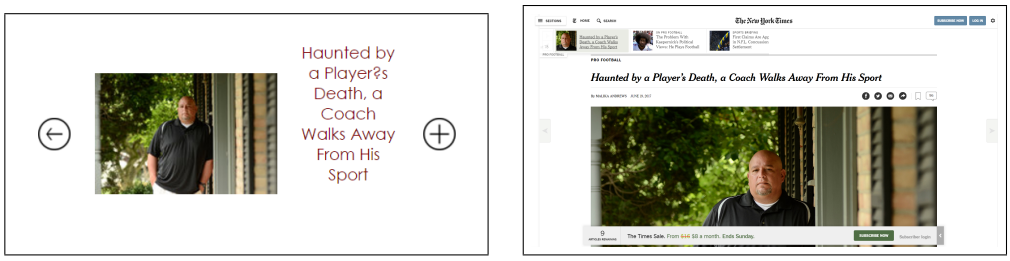

Figure 3.2: (a) Item block (b) News article.

their desired item, like the link displayed in Figure 3.2(a), before they quit the interface system. Whenever users hit the link, a new page with the target information will jump out from the browser, like the one showed in Figure 3.2(b).

At this point, we can conclude that a comprehensive information retrieval process is accomplished by an interactive retrieval system. However, some major weakness is revealed during the process. First of all, to support adaptive web design in an interactive retrieval system, the controller needs to detect screen size and design the layout adaptively. In the previous prototype, there is no channel for screen-display related parameters to transfer. Furthermore, the major three parameters, including *size*, *app* and *algo*, are stationary in each URL. It becomes a problem while users expect to access different sized interface. They have to manually modify the URL which is very inconvenient.

CHAPTER 4

SYSTEM IMPLEMENTATION

In the above chapter, we provide an overview of system architecture for the previous prototype interface system. Besides, we discuss the weakness from the perspective of Model, View and Controller. In this chapter, we will present how AdaptiveScreen overcomes these flaws and show how we apply adaptive web design techniques on it. For the purpose of offering a better understanding on the system, we hand over a diagram of MVC architecture with logic flow. In addition, we list all the highlighted functions and challenges involved as well as how we addressed them.

To conceive a good architecture that is maintainable in time, we need to clarify what key features we expect. First of all, the interface system should be responsive in both websites and mobile platforms. In other words, the layout of item and tag blocks need to be automatically adjusted depended on the screen layout including size and direction. Secondly, we assume the project to be multi-pages and performant. All the button should be selectable and the web pages are supposed to jump into another lap of the system or navigate user to the desired news article quickly. Furthermore, the new interface should be compatible with the back-end and similar to the previous design because AdaptiveScreen aims to flourish the power of Navigation Card Model and should take care of the users who are already familiar with the prototype. Last but not least, we want to reuse the same design pattern for any future projects related to Navigational Card Model. Strictly speaking, the implementation of AdaptiveScreen will be as independent as possible to previous works. If the authors of paper [1] make any further improvements on the server side, there should be no obstacle for them to rebuild AdaptiveScreen.

Besides pointing out key features, we also need to organize the work-flow. In most front-end projects, there are lots of different types of files, such as PHP, JavaScript and CSS files. To set up a good front-end work-flow, we demand a way to manage these different technologies. Therefore, organizing

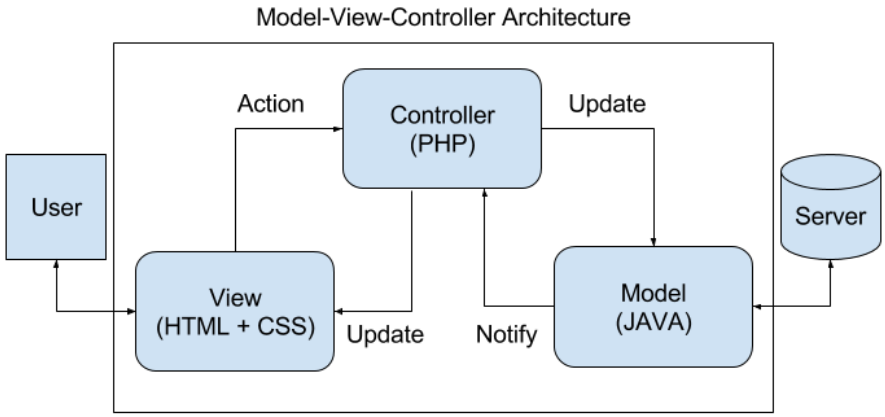

Figure 4.1: Model-View-Controller-High-Level-Diagram..

everything in folders, stick to a pattern or a convention in favor of keeping things clear and neat, is really important. In our case, all back-end code are developed in JAVA and stored separately. For AdaptiveScreen, the controller is established in PHP and JavaScript, and the view is set up in HTML and CSS.

After discussing the weakness of previous prototype and clarifying the good standards of interface design, we demonstrate the implementation details of AdaptiveScreen by illustrating highlighted functions and challenges we encountered during the development.

4.1 Detect screen size

We designed a simple JavaScript function *getSize* which can monitor the screen size by returning height and width. They are measured in the unit of pixel which can be directly applied for calculating CSS layout. The major challenge is to transmit these two parameters from web pages to the controller. With the inspiration of method that the previous prototype used to send parameters, we create two more parameters l and w in both PHP

and JavaScript files to represent height and width of observed web page. In specific, AdaptiveScreen will append two more parameters at the tail of original URL. However, at this point, these two parameters cannot be updated spontaneously during the interactive information retrieval process.

4.2 Calculate number of cards to display

With the help of function *getSize*, we have the ability to measure height and width of web pages. At the same time, the dimension of card block is defined in advance and remains unchangeable during the whole interacting process. Hence, the problem of calculating number of cards to display according to the screen sizes can be solved by simple math. The rest of challenges are about typesetting and card layout arrangement. On the one hand, one of the main contributions of Navigation Card Model is to prove that the necessity of scrolling action can be get rid of. Therefore, we need to let the server know how many card blocks to generate, no more or no less. In other words, the number of card block should just fit in the whole web pages. On the other hand, the CSS file served for previous prototype can only deal with one card block. In other words, we need to design a mechanism which can dynamically modify the distance between tags block and the outer frames so that the outer frame can always fit in no matter how many card blocks are generated.

4.3 Adaptive web design

After setting up functions *getSize* and *getCardNum*, we can finally synchronize the new AdaptiveScreen interface with Navigation Card Model. Since we are targeting on an adaptive web design interface, the final system operates in the following ways.

4.3.1 HTTPRequest

Whenever users type in the URL of our system into any web browser through PC or mobile phones, the adaptive web design mechanism will detect height

and width of web pages in HTML and send those parameters back to the controller which is handled in JavaScript through XMLHttpRequest.

4.3.2 Card generation

After the PHP controller receives web page information, we calculate how many number of card blocks to display and how big the outer frame should be. In the meantime, a card generation request is sent to the back-end and a CSS updating command is sent to the front-end.

4.3.3 Loading the initial page

Due to the natural layout of card blocks and a careful consideration on usage habits of mobile devices customers, we decide to only support vertical expansions among all interfaces. In other words, only the alteration in height of the screen can affect layout of the interface adaptively. The following use cases on laptops and mobile phones are offered to readers for better understanding our system.

Access through desktop web browser

Suppose users open the web browser in full screen on a HP Envy 15 laptop with 15.6 inches screen, the full resolution on a Chrome web browser is detected as 742*1419 pixels where the front number represents the height and the later one reflects the width. According to the default typesetting of card blocks, each single tags block occupies a range of 180*320 pixels. Please note that the next button and back button do not belong to block content and they have exactly the same height as the tags block. According to our plan on only supporting vertical expansion, we have no need to consider the width of both next button and back button. Therefore, users meet four card blocks in total on the initial page illustrated in Figure 4.2. This result is reasonable because the floor of 742 divided by 180 is four.

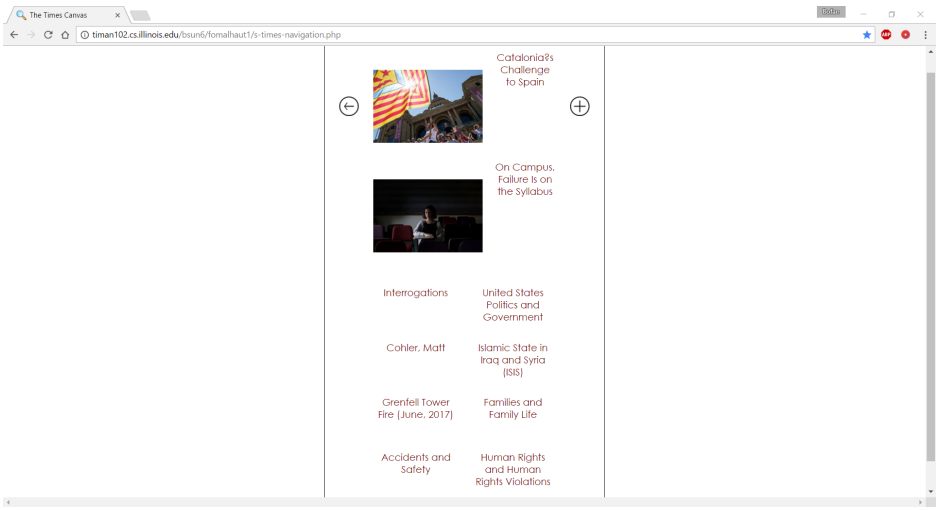

Figure 4.2: AdaptiveScreen on laptop web browser in full screen mode.

Access through mobile devices

Assume users have mobile devices like an iPhone 6S, when they reach our website, they will find totally seven cards presented in Figure 4.3. According to the return value of function *getSize*, the resolution of an iPhone 6S screen in vertical mode is 1409*980 pixels. Ideally, the system should display the floor of 1409 divided by 180 which is seven number of card blocks. This result matches the example displayed in Figure 4.3. At the moment, we can make the conclusion that the initial loading step can provide automatically layout adjustment on both laptops and mobile devices.

4.3.4 Automatic layout adjustment

After the initial page is loaded successfully, users may drag the web browser on a PC or rotate the mobile devices to change the screen size or direction. Or, users are satisfied with current screen expansion and they will formally enter the interactive information retrieval process. Actually, these two scenarios are very common but totally different. And lots of challenging questions raise on how to integrate adaptive web design and Navigation Card Model, such as (a) what are the system supposed to do if the screen gets smaller? Refreshing the system to a new page or deleting some card blocks? If do so, which card blocks should we delete? Starting from the top or the bottom? What if the screen gets larger? Refreshing or asking the

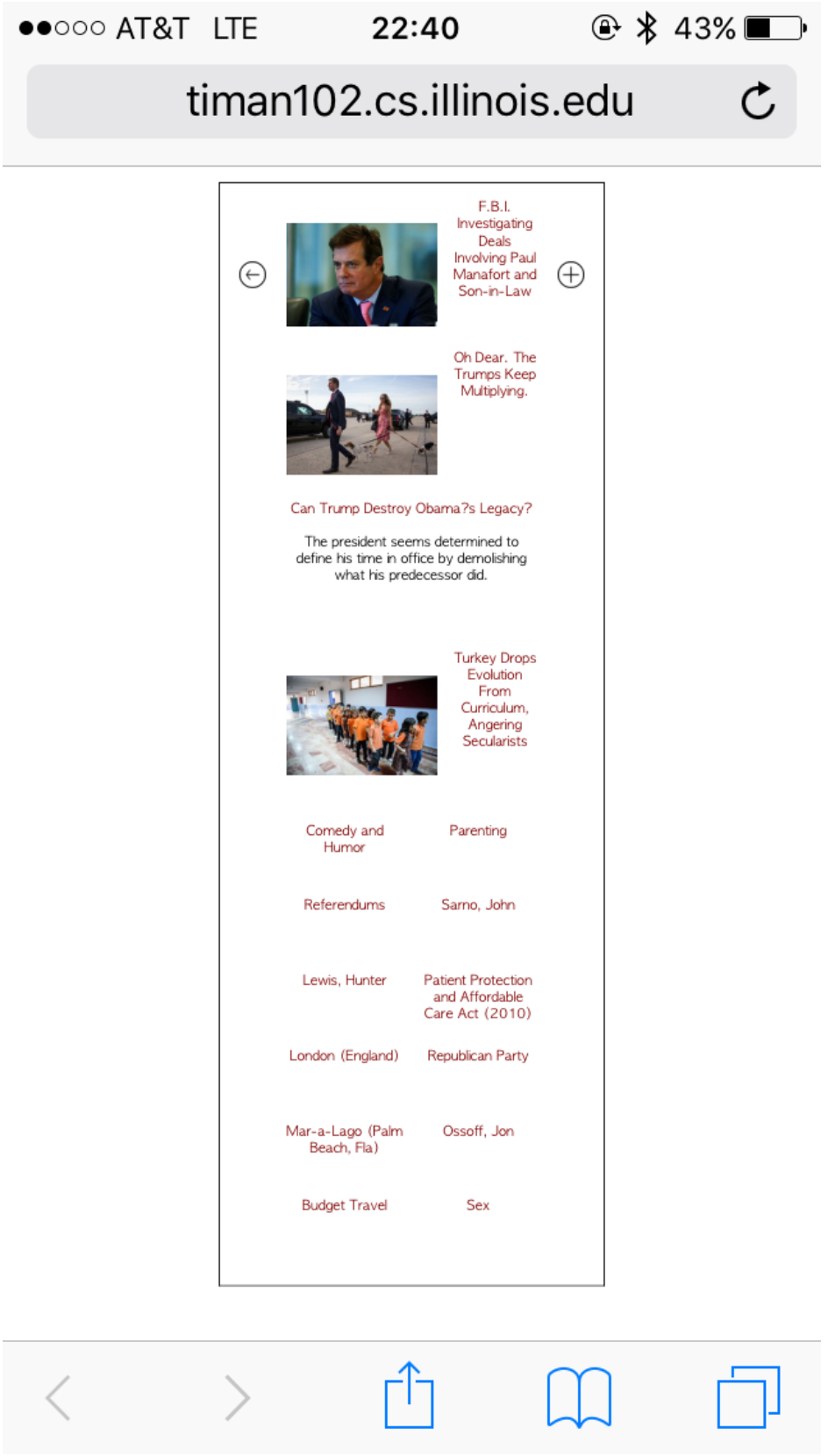

Figure 4.3: AdaptiveScreen on iPhone 6s in vertical screen mode.

server for more cards? (b) Another great challenge may happen during the process. What should the interface react when the user has already entered the interactive information retrieval process? Terminating and resetting to the initial lap? Is there a way to avoid this and continue the process with more or less cards? (c) In the end, the most challenging part is to answer the question that how we can distinguish the action of shifting screen size and the action of interacting with system. Both actions need the system to offer new page contents. How can we let the system know which reaction should it perform?

With a plenty of experiments and use case simulations, we address these challenges in following steps. (a) First of all, we decide to refresh the whole system and reset it to the initial page once users make any changes on number of card blocks, no matter which stage users are positioned in the interacting process. Here is one essential reason which can explain our decision. Customers tend to adjust the screen size before entering the system instead of during the process. Once they correct the web page range and begin the procedure, there is very few motivation for them to adjust the screen again since clicking all the tags and buttons will not alter the perfect-conditioned interface layout which are set by them at the very first place. (b) Secondly, if users have already begun their interaction with the Navigation Card Model and then choose to alter the screen, we have to terminate the process and reset the whole system. The reason is that removing or increasing any card blocks during the process will violate the assumptions and definitions of the Navigation Card Model. For example, according to the selection action assumption, in each lap, the user could either click a tag on the card or select *next* button. If we choose to simply retrieve more card blocks in current lap, this action will affect Preference which is defined as the systems estimated probability distribution characterizing the users interest in each item. (c) In the end, distinguishing between selecting actions and layout adjusting actions is a technical problem. We need to let the system know which kinds of action the user made and which reaction the system should perform. Similar to the communicating mechanism applied in function *getSize*, we use another parameter *type* to tell system the type of action made by customers. If users only select tags or buttons, the controller will receive a value *button* from parameter *type* after each actions. In the controller, there is an if-statement which executes the main Navigation Card Model functions normally when

the parameter *type* is parsed with value *button*. Otherwise, the controller considers that a screen adjustment is triggered and will use the if-statement to opt-out the main function so that a refreshing mechanism resets pages into initial lap with pure new card contents.

4.3.5 Exit the system

In the end, the users will either reach an item block they are looking for or in some other case, they may terminate the process and quit the system. All item blocks are connected with news articles in hyperlinks and will navigate customers to the websites where the new articles originally belong to.

CHAPTER 5

EVALUATION

AdaptiveScreen aims to support adaptive web design and further enhance the user experience over the prototype designed for Navigation Card Model. Therefore, we will compare with previous interface as the baseline. Furthermore, we identify successful cases where our new system is better and unsuccessful cases where it might be worse. In the end, some future research problems are suggested for researchers.

First of all, by comparing with Figure 5.1 and Figure 4.1, we safely conclude that the new interface system can fill in the web page with card blocks adaptively. In addition, the adaptive web design function is applied successfully when user shrink or enlarge screen sizes. In Figure 5.2(a), Figure 5.2(b) and Figure 5.2(c), we provide the screen shots on a laptop web browser with three, two and one card blocks displayed respectively. Moreover, to further test our system, we provide two more screen shots taken on an iPad with vertical and horizontal screen modes in Figure 5.3(a) and Figure 5.3(b). They display similar adaptive layouts as ones in iPhone 6S. Customers are benefited by this design since the more card blocks they have seen the more likely they may find target items. On the contrast, the previous design wastes lots of space on the web page.

In addition, we perfectly preserve the functionality of Navigation Card Model. Adaptive web design is an enhancement on the previous prototype, and therefore, the new design should not rebate any past works. By comparing the previous server website [1] and current AdaptiveScreen interface [2], we can tell that customers do not have any difficulties on learning and utilizing the new system.

Besides these accomplishments, we also meet some technical problems and dig out lots of interesting research questions during the development. One of

[1] http://timan102.cs.illinois.edu/yzhng103/fomalhaut/s-times-navigation.php
[2] http://timan102.cs.illinois.edu/bsun6/fomalhaut1/s-times-navigation.php

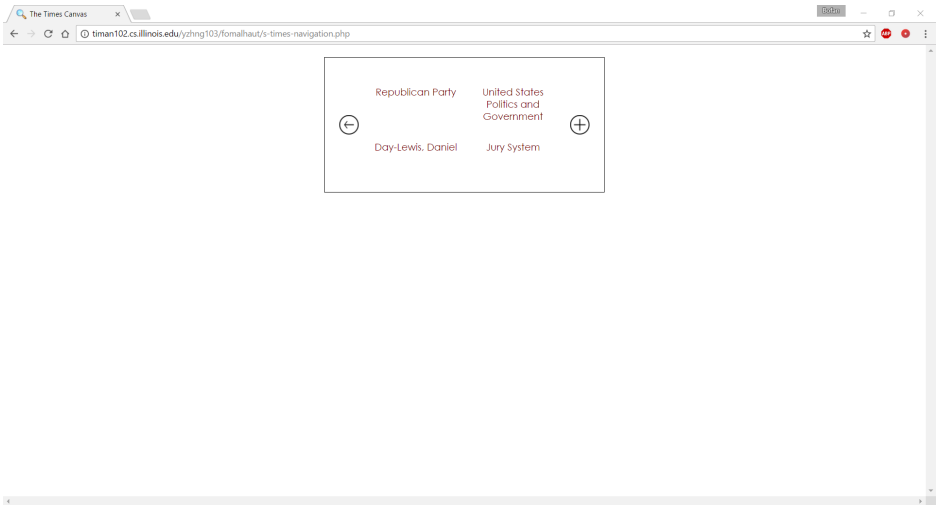

Figure 5.1: Prototype interface on laptop web browser in full screen mode.

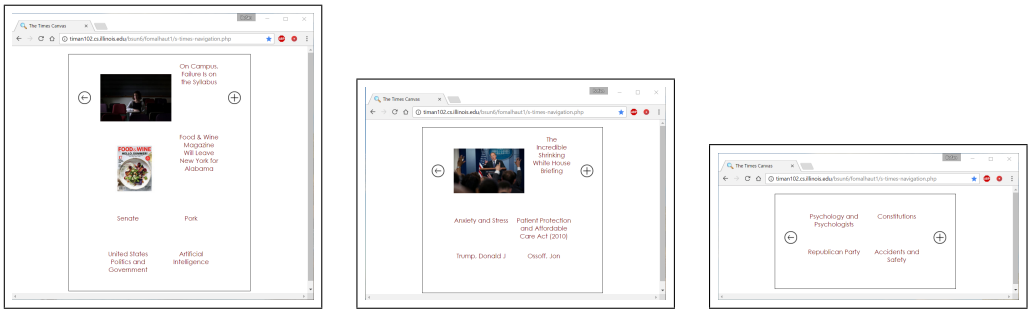

Figure 5.2: AdaptiveScreen interface on laptop with (a) three cards (b) two cards and (c) one card.

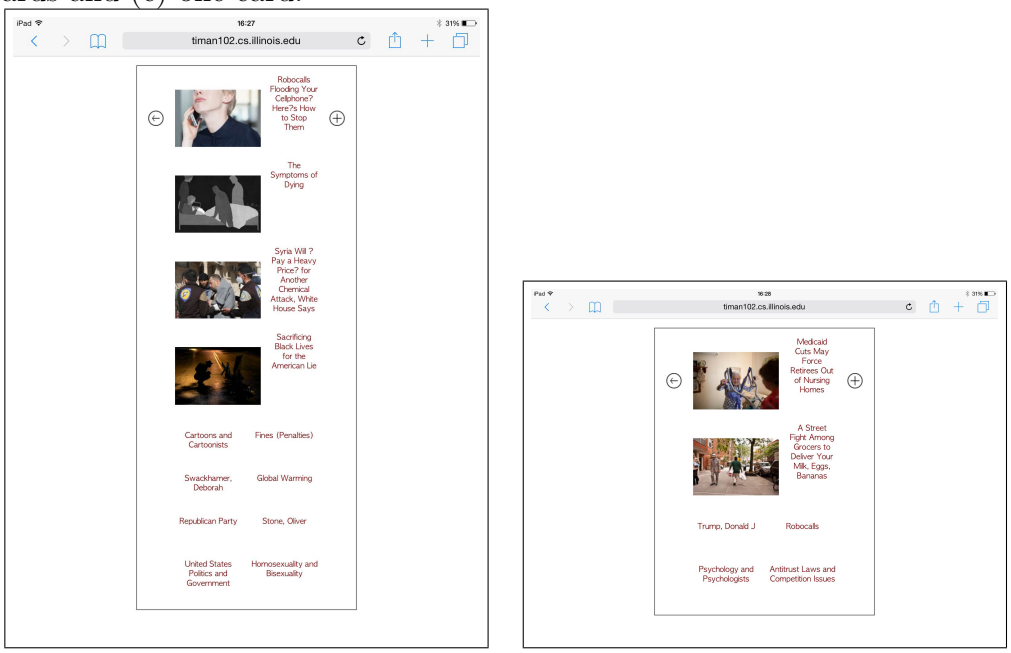

Figure 5.3: AdaptiveScreen interface on iPad2 in (a) vertical mode and (b) horizontal mode.

the major complaints from users is that the browser will refresh even though customers only drag a tiny range. Ideally, the interface is supposed to reset once there is a change on number of card blocks. We try to make it happen by using HTTP Cookies but failed.

For the future works, researchers may focus on implementing more mobile-based features. For example, since most smart phones have touch screen, we may use swipe gestures towards left and right to replace back and next button. In addition, the authors of Navigation Card Model also published another paper [2] which introduced a new definition called Stop tendencies. Stop tendencies is a measurement to assess the patience of users. By analyzing user behaviors, the refined model can automatically optimize interface layouts according to users stopping tendencies. One potential research direction is to further analyze search log data and utilize them with Adaptive-Screen so that it can benefit users during the interactive information retrieval process.

CHAPTER 6

CONCLUSION AND FUTURE WORK

We propose a novel adaptive interface system, AdaptiveScreen, which improves upon the previous prototype system based on ICM. To achieve the goal of building a browsing system with the functionality of ICM and adaptive web design, we analyze the system architecture, point out the strength and weakness, and implement several new functions. In particular, the system architecture is redesigned following by the rule of MVC and enhanced by advanced JavaScript controllers. The effectiveness of ICM is retained but the interaction between the model and the front-end is modified. Several new functions, including *getSize* and *getCardNum* are developed to support adaptive web design. In the end, we present plenty of screen shots of the prototype interface and the new AdaptiveScreen system. By comparing the layouts and performance on both laptops and mobile devices, we can conclude that AdaptiveScreen successfully supports the adaptive web design with a user-friendly interface. In addition, AdaptiveScreen makes a further step over the prototype system in the field of Human-Computer-Interaction (HCI) and Information Retrieval (IR).

REFERENCES

[1] Y. Zhang and C. Zhai, "Information retrieval as card playing: A formal model for optimizing interactive retrieval interface," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 685–694.

[2] Y. Zhang and C. Zhai, "A sequential decision formulation of the interface card model for interactive ir," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 85–94.

[3] N. Fuhr, "A probability ranking principle for interactive information retrieval," *Information Retrieval*, vol. 11, no. 3, pp. 251–265, 2008.

[4] N. J. Belkin, R. N. Oddy, and H. M. Brooks, "Ask for information retrieval: Part i. background and theory," *Journal of documentation*, vol. 38, no. 2, pp. 61–71, 1982.

[5] P. Ingwersen, "Cognitive perspectives of information retrieval interaction: elements of a cognitive ir theory," *Journal of documentation*, vol. 52, no. 1, pp. 3–50, 1996.

[6] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[7] S. E. Robertson and K. S. Jones, "Relevance weighting of search terms," *Journal of the Association for Information Science and Technology*, vol. 27, no. 3, pp. 129–146, 1976.

[8] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1998, pp. 275–281.

[9] D. Hiemstra and W. Kraaij, "Twenty-one at trec-7: Ad-hoc and cross-language track," 1999.

[10] C. Zhai, "Statistical language models for information retrieval," *Synthesis Lectures on Human Language Technologies*, vol. 1, no. 1, pp. 1–141, 2008.

[11] S. E. Robertson, "The probability ranking principle in ir," *Readings in information retrieval*, pp. 281–286, 1997.

[12] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Transactions on Information Systems (TOIS)*, vol. 7, no. 3, pp. 183–204, 1989.

[13] T.-Y. Liu et al., "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.